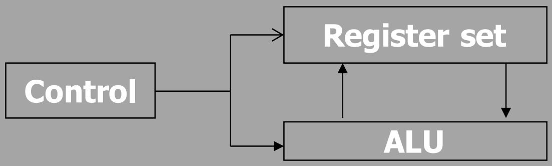
Processor Logic Design

Processor Organization



- The CPU is made up of 3 major Components.
- The CPU performs a variety of functions dictated by the type of instructions that are incorporated in the computer.
- In programming, memory locations are needed for storing pointers, counters, return addresses, temporary result , etc. Memory access is most time consuming operation in a computer.
- It is then more convenient and more efficient to store these intermediate values in processor registers, which are connected through common bus system.

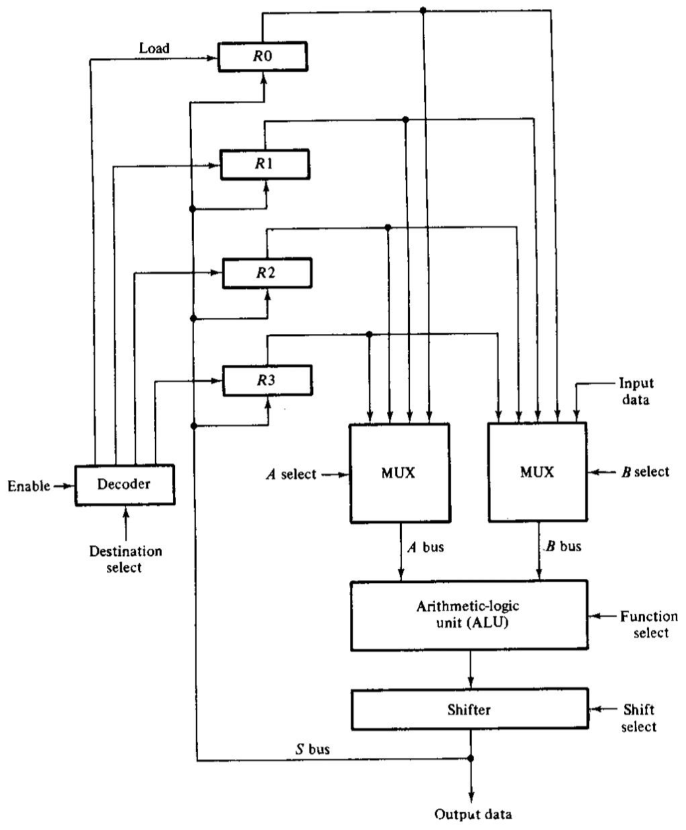
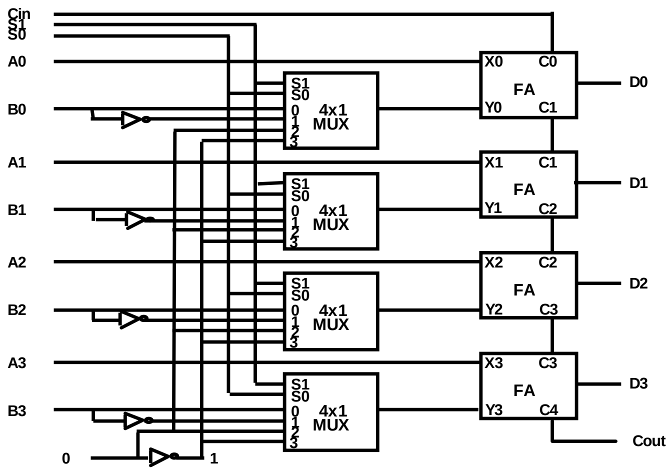


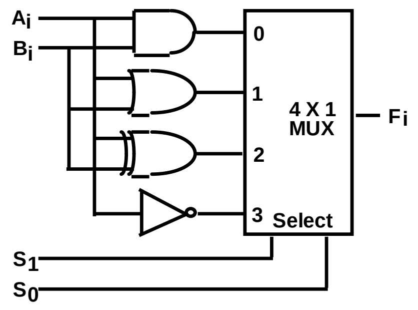
Figure 9-1 Processor registers and ALU connected through common buses

ARITHMETIC CIRCUIT



S1	S0	Cin	Y	Output	Microoperation
0	0	0	B	D = A + B	Add
0	0	1	B	D = A + B + 1	Add with carry
0	1	0	B’	D = A + B’	Subtract with borrow
0	1	1	B’	D = A + B’+ 1	Subtract
1	0	0	0	D = A	Transfer A
1	0	1	0	D = A + 1	Increment A
1	1	0	1	D = A - 1	Decrement A
1	1	1	1	D = A	Transfer A

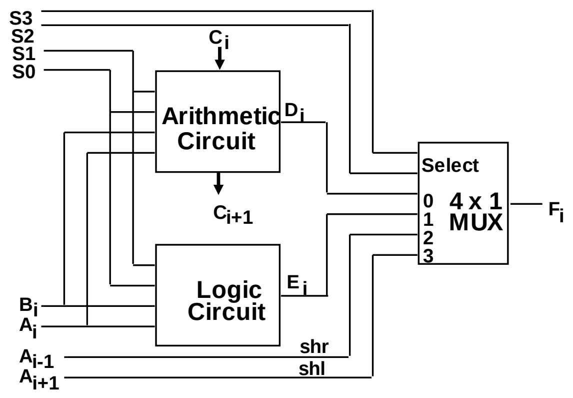
LOGIC CIRCUIT



Function table

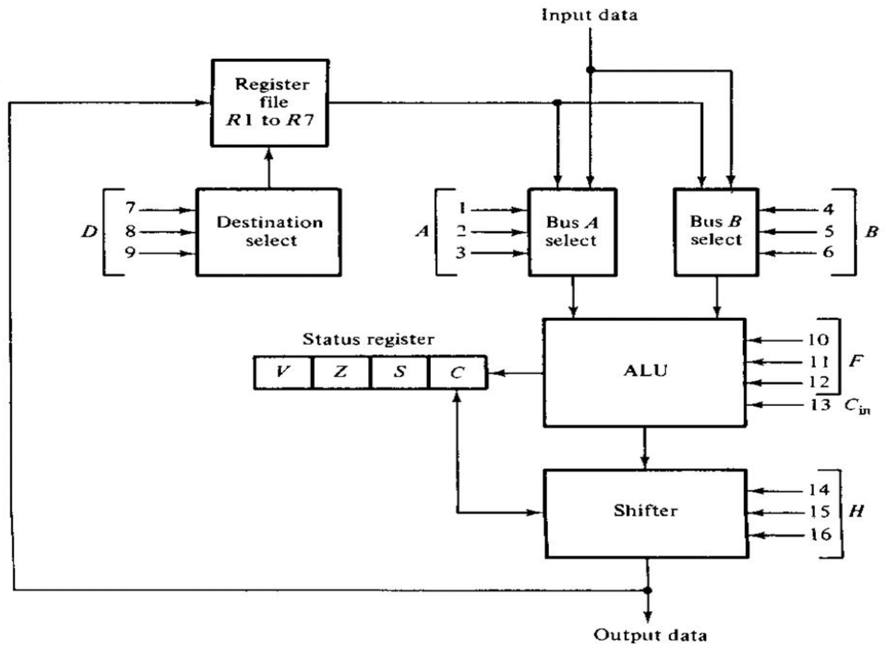
S_1	S_0	Output	μ-operation
0	0	$F = A \wedge B$	AND
0	1	$F = A \vee B$	OR
1	0	$F = A \oplus B$	XOR
1	1	F = A’	Complement

ARITHMETIC LOGIC SHIFT UNIT

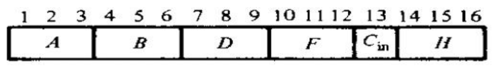


S3	S2	S1	S0	Cin	Operation	Function
0	0	0	0	0	F = A	Transfer A
0	0	0	0	1	F = A + 1	Increment A
0	0	0	1	0	F = A + B	Addition
0	0	0	1	1	F = A + B + 1	Add with carry
0	0	1	0	0	F = A + B’	Subtract with borrow
0	0	1	0	1	F = A + B’+ 1	Subtraction
0	0	1	1	0	F = A - 1	Decrement A
0	0	1	1	1	F = A	TransferA
0	1	0	0	X	$F = A \wedge B$	AND
0	1	0	1	X	$F = A \vee B$	OR
0	1	1	0	X	$F = A \oplus B$	XOR
0	1	1	1	X	F = A’	Complement A
1	0	X	X	X	F = shr A	Shift right A into F
1	1	X	X	X	F = shl A	Shift left A into F

Processor unit



(a) Block diagram

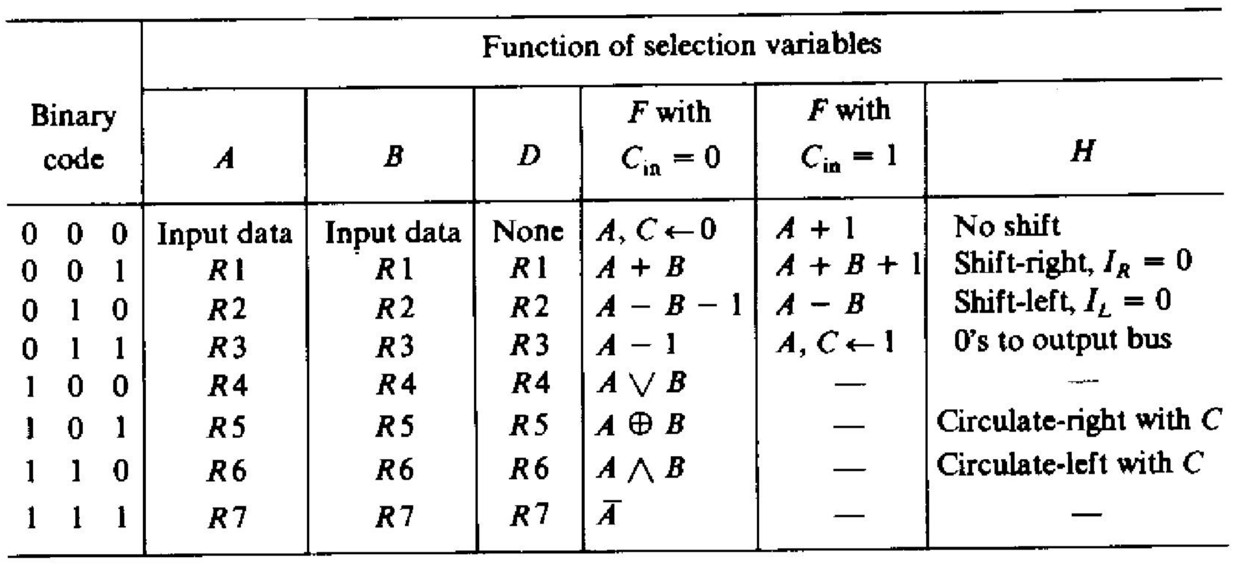


(b) Control word

Figure 9-16 Processor unit with control variables

Processor unit

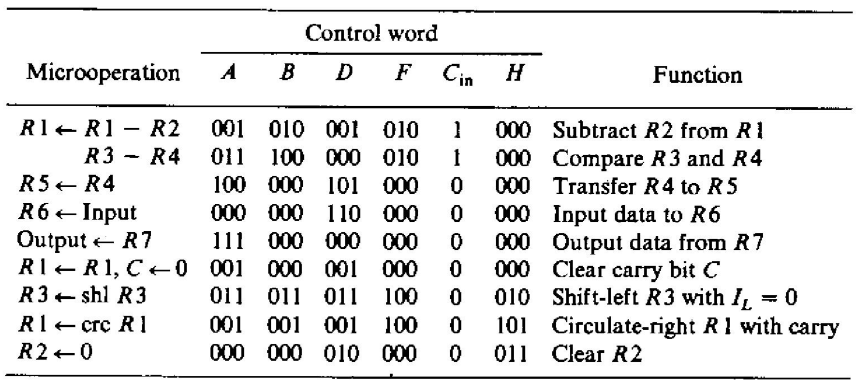
TABLE 9-8 Functions of control variables for the processor of Fig. 9-16



	Function of selection variables					
Binary code	A	B	D	F with $C_{\text{in}} = 0$	F with $C_{\text{in}} = 1$	H
0 0 0	Input data	Input data	None	$A, C \leftarrow 0$	$A + 1$	No shift
0 0 1	$R1$	$R1$	$R1$	$A + B$	$A + B + 1$	Shift-right, $I_R = 0$
0 1 0	$R2$	$R2$	$R2$	$A - B - 1$	$A - B$	Shift-left, $I_L = 0$
0 1 1	$R3$	$R3$	$R3$	$A - 1$	$A, C \leftarrow 1$	0's to output bus
1 0 0	$R4$	$R4$	$R4$	$A \vee B$	—	—
1 0 1	$R5$	$R5$	$R5$	$A \oplus B$	—	Circulate-right with C
1 1 0	$R6$	$R6$	$R6$	$A \wedge B$	—	Circulate-left with C
1 1 1	$R7$	$R7$	$R7$	$\bar{A}$	—	—

Processor unit

Table 9-9 Examples of microoperations for processor



Microoperation	Control word						Function
	A	B	D	F	C_{in}	H	
$R1 \leftarrow R1 - R2$	001	010	001	010	1	000	Subtract $R2$ from $R1$
$R3 - R4$	011	100	000	010	1	000	Compare $R3$ and $R4$
$R5 \leftarrow R4$	100	000	101	000	0	000	Transfer $R4$ to $R5$
$R6 \leftarrow$ Input	000	000	110	000	0	000	Input data to $R6$
Output $\leftarrow R7$	111	000	000	000	0	000	Output data from $R7$
$R1 \leftarrow R1, C \leftarrow 0$	001	000	001	000	0	000	Clear carry bit C
$R3 \leftarrow$ shl $R3$	011	011	011	100	0	010	Shift-left $R3$ with $I_L = 0$
$R1 \leftarrow$ crc $R1$	001	001	001	100	0	101	Circulate-right $R1$ with carry
$R2 \leftarrow 0$	000	000	010	000	0	011	Clear $R2$

Computer Registers

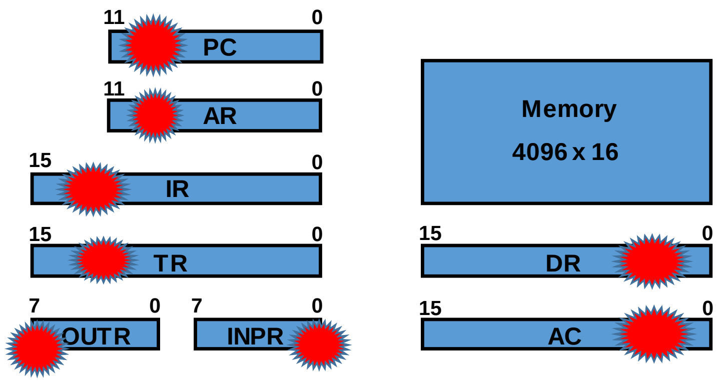
- Computer instructions are **normally stored** in **consecutive** memory locations and **executed sequentially one at a time**
- The **control reads** an instruction from a **specific address** in memory **and executes it**, and so on
- This type of sequencing needs a **counter** to calculate the address of the **next instruction** after execution of the current instruction is completed

Computer Registers $^{\text{cont.}}$

- It is also necessary to provide a register in the control unit for storing the instruction code after it is read from memory
- The **computer needs** processor **registers** for **manipulating data** and a **register** for **holding a memory address**

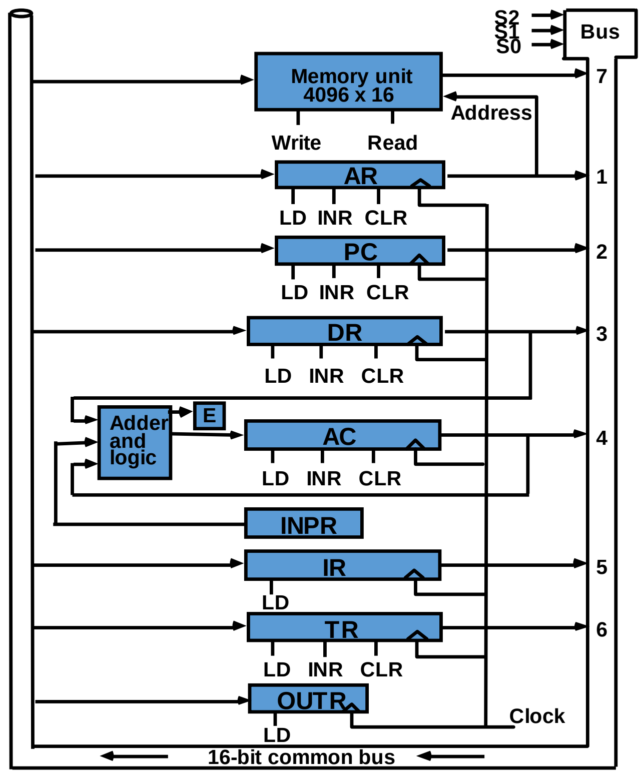
In order to cover the basic concepts behind designing a computer, a model (an imaginary system) will be presented to you throughout this chapter. This model will be called the "Basic Computer"

Registers in the Basic Computer



List of Basic Computer Registers

DR	16	Data Register	Holds memory operand
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds input character
OUTR	8	Output Register	Holds output character



Computer Registers Common Bus System

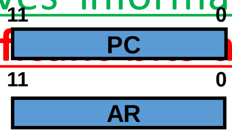
Computer Registers
Common Bus System cont.

- **$S_2S_1S_0$**: ***Selects the register/memory that would use the bus***
- **LD (load):** ***When enabled, the particular register receives the data from the bus during the next clock pulse transition***
- **E (extended AC bit):** ***flip-flop holds the carry***
- ***DR, AC, IR, and TR: have 16 bits each***
- **AR and PC:** ***have 12 bits each since they hold a memory address***

Computer Registers

Common Bus System cont.

- When the contents of **AR or PC** are applied to the 16-bit common bus, **the four most significant bits are set to zeros**
- When **AR or PC** receives information from the bus, **only the 12 least signif[illegible]re transferred into the register**



- **INPR and OUTR:** communicate with the eight least significant bits in the bus

Computer Registers

Common Bus System cont.

- **INPR:** Receives a character from the input device (keyboard,…etc) which is then transferred to AC
- **OUTR:** Receives a character from AC and delivers it to an output device (say a Monitor)
- Five registers have three control inputs: ***LD (load), INR (increment), and CLR (clear)***
- ***Register $\equiv$ binary counter with parallel load and synchronous clear***

Computer Registers
Memory Address

- ***<u>The input data and output data of the memory are connected to the common bus</u>***
- But the **memory address** is **connected to AR**
- Therefore, AR must always be used to specify a memory address
- *By using a single register for the address, we eliminate the need for an address bus that would have been needed otherwise*

Computer Registers Memory Address cont.

to

- **Register → Memory:** Write operation
  to
- **Memory → Register:** Read operation (**note that AC cannot directly read from memory!!)**
- ***Note that the content of any register can be applied onto the bus and an operation can be performed in the adder and logic circuit during the same clock cycle***

Computer Registers
Memory Address cont.

- **The transition at the end of the cycle transfers the content of the bus into the destination register, and the output of the adder and logic circuit into the AC**
- For example, the two microoperations

  DR←AC and AC←DR (Exchange)

  can be executed at the **same time**
- **This is done by:**

Computer Registers

Memory Address cont.

- ✓1- ***place the contents of AC on the bus ($S_2S_1S_0$=100)***
- ✓2- ***enabling the LD (load) input of DR***
- ✓3- ***Transferring the contents of the DR through the adder and logic circuit into AC***
- ✓4- ***enabling the LD (load) input of AC***
- **All during the same clock cycle**
- The two transfers occur upon the arrival of the clock pulse transition at the end of the clock cycle